Scalable Performance Diagnostics on Leadership Computing Platforms

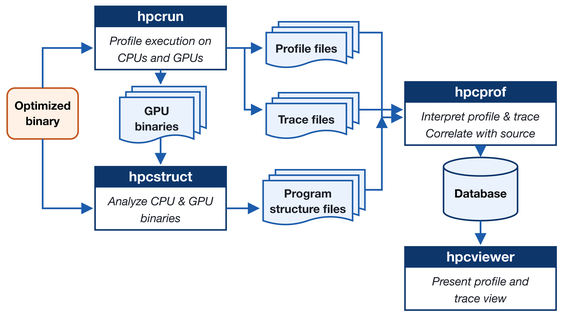
Dragana Grbic
Department of Computer Science, Rice University

1 Motivation

- **Exascale supercomputers** at ANL and ORNL let users run executions across **thousands of compute nodes**
- Massive parallel execution contexts: numerous **MPI ranks**, **CPU threads**, and **GPU streams**
- Interpreting performance at this scale remains an **open problem** in the HPC community

2 Measuring Applications at Large Scale

HPCToolkit is a powerful measurement tool that uses statistical sampling of hardware counters and call stack unwinding.



- Fine-grained, instruction-level measurement of CPU and GPU code with minimal overhead
- Profiles and traces applications across thousands of compute nodes

But GUI-based analysis at exascale is tedious and time-consuming

- ✕ No support for automation
- ✕ No support for scalable, efficient, interactive analysis
- ✕ No support for integration with external tools

★ Our Contributions

Beyond refining ***hpcanalysis***, three novel enhancements:

1. A novel algorithm that scans the summary profile **once** to detect critical code regions and bottlenecks, then propagates the extracted indices across all MPI ranks, CPU threads, and GPU streams via indexing, binary search, and internal parallelism
2. A novel ***HpcDataFrame*** abstraction for vendor-neutral GPU-accelerated processing on NVIDIA (*cuDF*), AMD (*hipDF*), and Intel (custom SYCL wrapper) GPUs, replacing naive Pandas CPU-based processing
3. A novel **three-dimensional Thicket model** that "re-materializes" iterative behavior from execution traces, enabling analysis of iterative patterns and load imbalances within and across traces

100K — MPI ranks profiled on Aurora (1,000 compute nodes)

9.69 s — to ingest all 100K profiles

314.54× — peak GPU speedup (grouping, Polaris)

32.28% — predicted speedup for GAMESS on Frontier

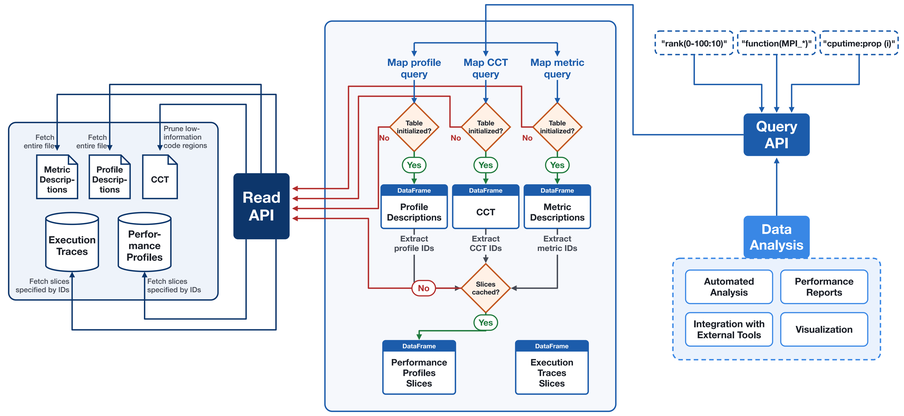
3 Refining *hpcanalysis* for Large-Scale Performance Analysis

We refined ***hpcanalysis*** into a **multi-layered infrastructure**:

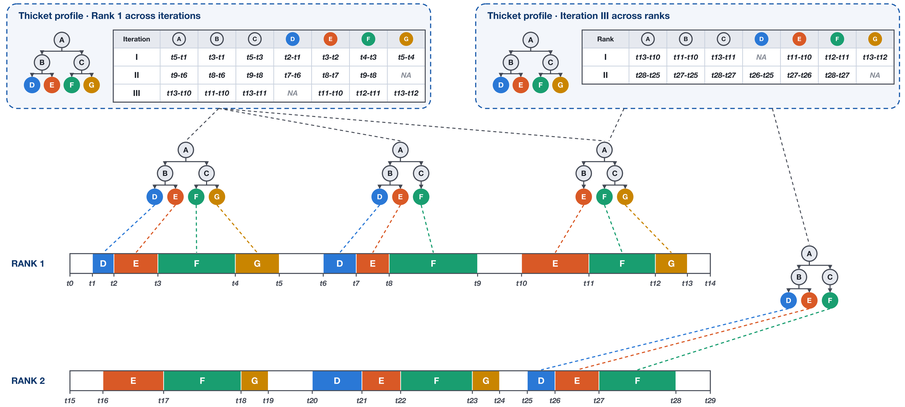
Read API · C++
High-throughput ingestion and selective access of performance data

Query API
Specialized query expressions and GPU-accelerated processing on NVIDIA, AMD, and Intel GPUs

Python API
Automation and interactive analysis; can be integrated with AI frameworks for automated analysis



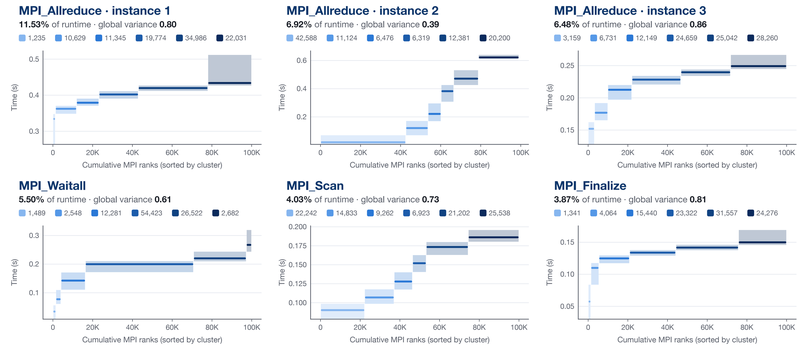
4 Advanced Trace Analysis



- ***Thicket*** is a tool developed at LLNL for two-dimensional, cross-experiment analysis of performance profiles
- We integrated ***hpcanalysis***' interface with Thicket and extended it into a **three-dimensional model** that "re-materializes" iterative behavior from execution traces
- A 3-D DataFrame index (**calling context**, **trace index**, **iteration index**) lets users examine iterative patterns and load imbalances with a few lines of Python
- With this model, we detected and predicted a potential **32.28% speedup** under ideal GPU load balancing for a GAMESS workload on Frontier

5 Processing Massive-Scale Executions

- AMG on **Aurora**: 1,000 compute nodes × 100 MPI ranks per node, producing **100K parallel profiles**, each with up to 97,833 distinct calling contexts
- ***hpcanalysis*** pruned the summary profile and propagated the extracted indices across all 100K ranks, ingesting the entire dataset in **9.69 seconds**
- Queries `"rank"`, `"function(MPI_*)"`, and `"cputime:prop (i)"` reveal load imbalance across MPI bottlenecks



- Clustering uncovered a critical *MPI_Allreduce* instance with severe imbalance, split across **798 and 202** compute nodes
- Mapping node IDs to Aurora's interconnect addresses localizes outliers to their physical **rack and chassis** in the Slingshot network
- Explores a massive-scale execution **in seconds**, detecting critical bottlenecks, outliers, and congestion

6 Benchmarking *HpcDataFrame*

- Processed execution traces from the 100K-rank AMG run, comparing the GPU backend with the Pandas CPU backend on Polaris, Frontier, and Aurora
- CPU often wins on small datasets (10–100 traces); at scale, speedups reach **314.54×** (grouping, Polaris), **220.10×** (sorting, Frontier), and **81.71×** (cumulative sum, Aurora)

CPU (Pandas) faster — 0.05× — 1× — 10× — 100× — GPU (HpcDataFrame) faster

Polaris · NVIDIA A100 · cuDF

Traces	Grouping	Sorting	Filtering	Merging
10	1.70×	0.40×	0.10×	3.92×
100	3.05×	3.37×	0.40×	3.73×
1,000	15.15×	11.15×	1.08×	5.27×
10,000	121.13×	111.44×	8.55×	41.99×
100,000	314.54×	209.40×	42.18×	67.11×

Frontier · AMD MI250X · hipDF

Traces	Grouping	Sorting	Filtering	Merging
10	0.82×	0.46×	0.24×	4.53×
100	3.59×	3.62×	0.42×	5.57×
1,000	20.84×	28.22×	1.82×	11.55×
10,000	79.51×	132.39×	23.38×	54.41×
100,000	154.23×	220.10×	123.75×	83.70×

Aurora · Intel Max 1550 · SYCL

Operation / traces	10	100	1,000	10,000	100,000
Scalar compare	0.35×	0.44×	2.01×	8.28×	11.50×
Vector add	0.42×	0.51×	4.76×	1.49×	8.93×
In-place multiply	0.61×	0.78×	6.93×	25.34×	63.11×
Sorting	0.27×	2.13×	12.75×	23.31×	23.47×
Filtering	0.05×	0.19×	1.74×	4.92×	59.29×
Reduction sum	0.12×	0.26×	1.88×	12.46×	68.49×
Cumulative sum	0.18×	1.35×	9.82×	14.58×	81.71×